01 Jun 2003

A Software Debugger Interface for an 8051 Hardware Model

Lokesh Verma

Hardy J. Pottinger
Missouri University of Science and Technology, hjp@mst.edu

Daryl G. Beetner
Missouri University of Science and Technology, daryl@mst.edu

Recommended Citation

L. Verma et al., "A Software Debugger Interface for an 8051 Hardware Model," *Proceedings of the IEEE International Conference on Microelectronic Systems Education (2003, Anaheim, CA)*, pp. 112-114, Institute of Electrical and Electronics Engineers (IEEE), Jun 2003.
The definitive version is available at https://doi.org/10.1109/MSE.2003.1205279

A Software Debugger Interface for an 8051 Hardware Model

Lokesh Verma, Hardy J. Pottinger, and Daryl G. Beetner
Department of Electrical and Computer Engineering
University of Missouri-Rolla
1870 Miner Circle, Rolla, MO 65409-0040
{verma,hjp,daryl}@umr.edu

Abstract

A VHDL model of the 8051 microcontroller is a key component for a course in hardware-software codesign, in its second year of development at University of Missouri-Rolla. Our paper discusses a software-centric user interface developed for this model using Tcl/Tk. Preliminary experience with using the debugger in an undergraduate laboratory is discussed.

1. Introduction

The University of Missouri-Rolla offers a microelectronic systems course featuring embedded systems and hardware-software co-design as well as an associated laboratory. The laboratory course is being developed under a grant from the National Science Foundation (NSF) as a series of laboratory experiments teaching fundamentals of microcontrollers, digital systems, and hardware-software co-design through hands-on development of simple embedded systems [1, 2]. The laboratories are built around the 8051 microcontroller and the Xilinx Field Programmable Gate Array (FPGA). The areas of hardware-software co-design and embedded systems covered by the laboratories include:

- Hardware-software co-development
- Hardware-software co-verification
- Partitioning tasks between hardware and software
- Re-use of IP
- Interfacing the 8051 with external devices
- Implementation of digital logic in an FPGA
- Programming microcontrollers in assembly and C

A key component to the development of the laboratories is a VHDL model of the 8051 microcontroller. A board-level model that combines the 8051 model with an SRAM model that can read Intel Hex files, a seven-segment display model, and a Xilinx XC4005XL FPGA was also developed. These models enable cosimulation of hardware and software using Mentor Graphics' QuickHDL Pro and a mixture of schematic and HDL models. Through simulation, students verify the operation of their hardware design along with software. Once the design is working properly in simulation, the design is verified in hardware using an XS40 board from Xess Corporation.

Students develop their software in C or assembly-level language using Keil Software's software development tools. Before implementing their software with hardware, students completely simulate their software using Keil's software simulation tools. These tools are good for discovering high-level problems with the code, but not for simulating the hardware-software interface. After simulation, their code is converted to an Intel-format hex file for the 8051. They develop custom digital hardware to be implemented in the FPGA using Mentor Graphics' Design Architect.

During co-simulation of hardware and software in the QuickHDL Pro environment, students do not have access to the internal signals of the 8051 VHDL model. While simulating, they neither have access to the program nor can they step through the program one instruction or one clock-cycle at a time. It is very difficult for them to simulate to a particular point in the program and analyze the intermediate results. Detailed control of the simulation would require more knowledge about VHDL than the students currently possess.

To resolve these problems, a more familiar graphical user interface (GUI) debugger was developed using the Tcl/Tk scripting language. Although not intended to replace a more traditional software debugger, the new interface is a much more 'software-centric' way to control a combined hardware/software simulation. The GUI debugger displays the internal signals of the 8051 model and provides students the facility to step through their 8051 program one instruction cycle or one clock cycle at a time while simultaneously simulating their hardware. Breakpoints can be set in the user's machine code to analyze intermediate results. The GUI debugger also allows internal and external memory of the 8051 model to be viewed easily.

2. Hardware-Software GUI Debugger

The GUI debugger is designed using Tcl/Tk extensions to the ModelSim component of QuickHDL Pro. ModelSim, the VHDL simulator, comes with a built-in Tcl/Tk interpreter. The debugger uses the ModelSim 'when' command to access all of the signals inside the 8051 model. Whenever a signal changes value, the corresponding Tcl variable is automatically updated. Using the 'trace' command of the Tcl language, a procedure is automatically run every time a Tcl variable changes.

The GUI debugger, as shown in Figure 1, comprises of a main window and many sub windows that can be invoked from the menu bar of the interface. The main interface displays the contents of frequently referenced registers like

Proceedings of the 2003 IEEE International Conference on Microelectronic Systems Education (MSE'03)
0-7695-1973-3/03 $17.00 © 2003 IEEE

r0-r7, accumulator, program counter, data pointer, and stack pointer. It also shows the current state of the 8051 machine cycle, the currently executing instruction, and others. The assembly-level program and associated machine code are also listed on the main interface. The assembly-level program is reconstructed from the hex file provided to the VHDL 8051 model. Each instruction is preceded by its address in memory.

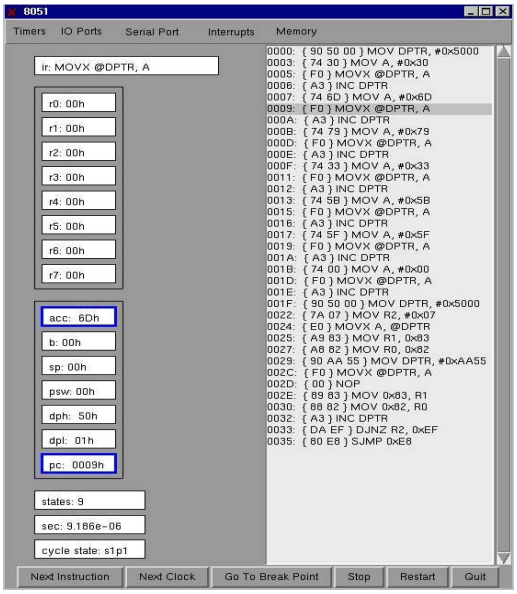

Figure 1. GUI Debugger

The user can optionally bring up additional windows to view information about timers, I/O ports, interrupts, the serial port, or memory with the menu bar provided. The 8051 VHDL model has internal code, internal data and external memory, which are easily accessible from the memory menu button. The user can specify a range of memory locations in hexadecimal format to view its contents. The debugger also allows user to step through the program one instruction or one clock cycle at a time with the help of the buttons provided at the bottom of the interface. The user can set breakpoints in the code to analyze intermediate results. The 'Stop' button can be used to halt the execution at any point. The 'Restart' button restarts the simulation. The 'Quit' button can be used to exit the ModelSim environment. Whenever a register changes its value the box containing the register value is highlighted in blue. The instruction to be executed next is also highlighted in the code-list box.

3. Classroom usage of the GUI debugger

A laboratory exercise was modified to use the hardware-software GUI debugger. In one of the laboratories students write an assembly-level program to partition the 8051's external memory into data and code segments. The data segment holds a message that is displayed on a seven-segment display which is mapped to an external memory location. The program is responsible for initializing the message table created in the data segment, for retrieving characters from the table, and then writing the characters to the seven-segment display with a short delay between consecutive characters.

Before the 8051 hardware debugging GUI, students debugged the software portion of their hardware-software design using Keil µVision 2. Their hardware and software were debugged together in QSPro, but they had few tools to control the simulation and observe what was happening in the 8051. With the use of the new GUI debugger, students were able to more easily control the simulation of the software and hardware (the seven-segment display, external memory) combination. They used the interface to view the internal signals of the 8051 and particular locations in memory (data and code segments). By simulating their program instruction-by-instruction, or clock-by-clock, or by setting breakpoints they had more control over the debug process.

4. Survey Results

A survey was conducted to evaluate the GUI debugger effectiveness in the laboratory. Preliminary results are encouraging. The majority of the students found the GUI debugger interface helpful in debugging their design. The debugger interface also helped them to better understand how the 8051 worked and how hardware and software worked together. Students felt time spent debugging their design was considerably reduced with the use of the debugger interface as they did not need to rely solely on signal traces to debug their design. The instructors teaching this lab noticed a significant improvement in students' performance and understanding of the lab compared to previous semesters.

5. Acknowledgement

This work was supported in part by the National Science Foundation's Course, Curriculum and Laboratory Improvement program under grant no. DUE-9952540.

6. References

[1]. D. G. Beetner, H. J. Pottinger, and K. Mitchel, "Laboratories Teaching Concepts in Microcontrollers and Hardware-Software Co-Design," *30th ASEE/IEEE Frontiers in Education Conference*, pp. SIC/1-5, 2000.

[2]. H. J. Pottinger and D. G. Beetner, "Hardware-Software Co-Verification in an Undergraduate Laboratory," *Proceedings 1999 IEEE Computer Society International Conference on Microelectronic Systems Education*, pp. 41-42, 1999.